In the Claims:

Please amend claims 1, 7, 17, 23, 29, 43 and 49. All pending claims are reproduced below, including those that remain unchanged.

1. (Currently Amended) A computer-implemented system to provide a common runtime container framework, comprising:

   a microprocessor;

   a plurality of runtime containers capable of processing service requests and providing application services, wherein the plurality of runtime containers are organized in a first hierarchical architecture;

   a plurality of metadata ~~object~~ objects capable of providing metadata on context, state, and/or other information about the data and objects being processed, wherein the plurality of meta objects are organized in a second hierarchical architecture; and

   ~~a hierarchical architecture capable of organizing the runtime container and the metadata object at levels within the hierarchical architecture~~

   wherein each runtime container of the plurality of runtime containers in the first hierarchical architecture can invoke service components within the runtime container and provide state information and context information to the service components based on a meta object of the plurality of metadata objects in the second hierarchical architecture at a corresponding level.

2. (Original) The system in claim 1, wherein:

   the runtime container is extensible via inheritance mechanisms, which inherit, provide, create and extend services, functionalities and properties of other runtime containers in the hierarchical architecture.

3. (Original) The system in claim 1, wherein:

   the metadata object is extensible via inheritance mechanisms, which inherit properties, methods and interfaces of other metadata objects in the hierarchical architecture.

4. (Original) The system in claim 1, wherein:

the runtime container and the metadata object are organized in duality, wherein a container at one level in a hierarchical architecture is capable of accessing a metadata object at the same level in the hierarchical architecture.

5. (Original) The system in claim 1, further comprising:

a well-defined API capable of creating new types of runtime containers, or customizing existing containers with incremental features.

6. (Original) The system in claim 1, further comprising:

a well-defined API capable of creating new levels in the hierarchical architecture for the runtime container and the metadata object.

7. (Currently Amended) A computer-implemented system to provide a common runtime container framework, comprising:

a microprocessor;

a routing and event handling component capable of communicating with an invocation component and adapted to be capable of communicating with external clients;

said invocation component capable of (1) receiving requests from the routing and event handling component, (2) dispatching said requests to a service component within a runtime container of a plurality of runtime containers, wherein the plurality of runtime containers are organized in a first hierarchical architecture, and (3) managing the returned responses;

at least one said service component within the runtime container capable of processing requests from the invocation component and producing responses back to the invocation component;

a state manager capable of obtaining state information from a nonvolatile storage and providing such information to the invocation component and the runtime container;

a context manager capable of obtaining information from a metadata object of a plurality of metadata objects and providing such information to the invocation component and the runtime container, wherein the plurality of meta objects are organized in a second

hierarchical architecture; ~~and~~

a control component capable of communicating with the runtime container and adapted to be capable of communicating with external services; and

wherein each runtime container of the plurality of runtime containers in the first hierarchical architecture can invoke service components within the runtime container and provide state information and context information to the service components based on a meta object of the plurality of metadata objects in the second hierarchical architecture at a corresponding level.

8. (Original) The system in claim 7, wherein:

the routing and event handling component is capable of communicating with the invocation component using a uniform or standardized protocol.

9. (Original) The system in claim 7, wherein:

the invocation component can be event-driven, wherein event delivery to a component and event generation from a component within the runtime container can be synchronous or asynchronous.

10. (Original) The system in claim 7, wherein:

the service component can be created in the form of Java Beans.

11. (Original) The system in claim 7, wherein:

the service component is capable of performing either pre-processing requests or post-processing of responses sent to or returned from the component.

12. (Original) The system in claim 7, further comprising:

a simplified component abstraction capable of transparently mapping the service component to a more complex set of components to generate and deploy applications under runtime environment.

13. (Original) The system in claim 7, further comprising:

a common configuration model capable of specifying the service component declaratively and programmatically, as well as providing a model for declarative configuration override of the component at application deployment time.

14. (Original) The system in claim 7, wherein:

the state manager is capable of locating, managing and persisting state information, wherein the physical mechanism of doing so is transparent.

15. (Original) The system in claim 7, wherein:

the context manager is capable of exposing component-level application services using the context information.

16. (Original) The system in claim 7, wherein:

the control component is capable of providing a simplified and common interaction model to communicate with the external services.

17. (Currently Amended) A computer-implemented system to provide a common runtime container, comprising:

a microprocessor;

at least one servlet capable of managing communications between the runtime container and external entities using common or uniform protocols;

at least one listener capable of monitoring incoming communication at the servlet;

a first dispatcher component capable of (1) communicating with one or more servlets, (2) determining which components to invoke, (3) dispatching requests requiring asynchronous processing to a queue, and (4) dispatching requests requiring synchronous processing directly to a stateful or a stateless component;

said queue capable of storing asynchronous requests;

a second dispatcher component capable of (1) receiving requests from the queue, (2) determining which components to invoke, and (3) dispatching requests requiring synchronous processing directly to a stateful or a stateless component;

at least one said stateless component capable of processing stateless requests; and

at least one said stateful component capable of processing stateful requests.

18. (Original) The system in claim 17, wherein:

the servlet is capable of communicating in TCP/IP, HTTP, SOAP, XML, and other application-specific protocols.

19. (Original) The system in claim 17, wherein:

the first or second dispatcher can be implemented using Java programming language in the form of Java Beans.

20. (Original) The system in claim 17, wherein:

the stateless or stateful component can be implemented using Java programming language in the form of Java Beans.

21. (Original) The system in claim 17, wherein:

the stateless component is capable of at least one of the following:

deriving context information from the metadata;

containing an arbitrary amount of code for processing logic;

calling other stateless components within the container; and

utilizing synchronous or asynchronous controls to communicate with external services.

22. (Original) The system in claim 17, wherein:

the stateful component is capable of at least one of the following:

deriving context information from the metadata;

retrieving state information from nonvolatile storage through a state management component;

containing an arbitrary amount of code for processing logic;

calling other stateless or stateful components within the container; and

utilizing synchronous or asynchronous controls to communicate with external services.

23. (Currently Amended) A method to provide a common runtime container framework, comprising:

processing service requests and providing application services via a runtime container of a plurality of runtime containers, wherein the plurality of runtime containers are organized in a first hierarchical architecture;

providing metadata on context, state, and/or other information about the data and objects being processed via a metadata object of a plurality of metadata objects, wherein the plurality of metadata objects are organized in a second hierarchical architecture; and

~~organizing the runtime container and the metadata object at levels within a hierarchical architecture~~

allowing each runtime container of the plurality of runtime containers in the first hierarchical architecture to invoke service components within the runtime container and provide state information and context information to the service components based on a meta object of the plurality of metadata objects in the second hierarchical architecture at a corresponding level.

24. (Original) The method in claim 23, further comprising:

extending the runtime container to inherit, provide, create and extend services, functionalities and properties of other runtime containers in the hierarchical architecture via inheritance mechanisms.

25. (Original) The method in claim 23, further comprising:

extending the metadata object to inherit properties, methods and interfaces of other metadata objects in the hierarchical architecture via inheritance mechanisms.

26. (Original) The method in claim 23, further comprising:

organizing the runtime container and the metadata object in duality, wherein a container at one level in a hierarchical architecture is capable of accessing a metadata object at the same level in the hierarchical architecture.

27. (Original) The method in claim 23, further comprising:

creating new types of runtime containers, or customizing existing containers with incremental features.

28. (Original) The method in claim 23, further comprising:

creating new levels in the hierarchical architecture for the runtime container and the metadata object.

29. (Currently Amended) A method to provide a common runtime container framework, comprising:

communicating with an invocation component and external clients via a routing and event handling component;

receiving requests from the routing and event handling component, dispatching said requests to a service component within a runtime container of a plurality of runtime containers, wherein the plurality of runtime containers are organized in a first hierarchical architecture, and managing the returned responses via the invocation component;

processing requests from the invocation component and producing responses back to the invocation component via the service component within the runtime container;

obtaining state information from a nonvolatile storage and providing such information to the invocation component and the runtime container;

obtaining information from a metadata object of a plurality of metadata objects and providing such information to the invocation component and the runtime container, wherein the plurality of meta objects are organized in a second hierarchical architecture; ~~and~~

communicating with the runtime container and external services; and

allowing each runtime container of the plurality of runtime containers in the first hierarchical architecture to invoke service components within the runtime container and provide state information and context information to the service components based on a meta object of the plurality of metadata objects in the second hierarchical architecture at a corresponding level.

30. (Original) The method in claim 29, further comprising:

communicating with the invocation component using a uniform or standardized protocol.

31. (Original) The method in claim 29, further comprising:

creating the service component in the form of Java Beans.

32. (Original) The method in claim 29, further comprising:

performing either pre-processing requests or post-processing of responses sent to or returned from the service component.

33. (Original) The method in claim 29, further comprising:

transparently mapping the service component to a more complex set of components to generate and deploy applications under runtime environment.

34. (Original) The method in claim 29, further comprising:

specifying the service component declaratively and programmatically, as well as providing a model for declarative configuration override of the component at application deployment time.

35. (Original) The method in claim 29, further comprising:

locating, managing and persisting state information, wherein the physical mechanism of doing so is transparent.

36. (Original) The method in claim 29, further comprising:

exposing component-level application services using the context information.

37. (Original) The method in claim 29, further comprising:

providing a simplified and common interaction model to communicate with the external services.

38. (Original) A method to provide a common runtime container, comprising:

managing communications between the runtime container and external entities using common or uniform protocols via at least one servlet;

monitoring incoming communication at the servlet;

communicating with one or more servlets, determining which components to invoke, dispatching requests requiring asynchronous processing to a queue, and dispatching requests requiring synchronous processing directly to a stateful or a stateless component;

storing asynchronous requests via the queue;

receiving requests from the queue, determining which components to invoke, and dispatching requests requiring synchronous processing directly to a stateful or a stateless component;

processing stateless requests via at least one stateless component; and

processing stateful requests via at least one stateful component.

39. (Original) The method in claim 38, further comprising:

communicating in TCP/IP, HTTP, SOAP, XML, and other application-specific protocols.

40. (Original) The method in claim 38, further comprising:

implementing the stateless or stateful component using Java programming language in the form of Java Beans.

41. (Original) The method in claim 38, wherein:

the stateless component is capable of at least one of the following:

deriving context information from the metadata;

containing an arbitrary amount of code for processing logic;

calling other stateless components within the container; and

utilizing synchronous or asynchronous controls to communicate with external services.

42. (Original) The method in claim 38, wherein:

the stateful component is capable of at least one of the following:

deriving context information from the metadata;

retrieving state information from nonvolatile storage through a state management component;

containing an arbitrary amount of code for processing logic;

calling other stateless or stateful components within the container; and

utilizing synchronous or asynchronous controls to communicate with external services.

43. (Currently Amended) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

process service requests and provide application services via a [[a]] runtime container of a plurality of runtime containers, wherein the plurality of runtime containers are organized in a first hierarchical architecture;

provide metadata on context, state, and/or other information about the data and objects being processed via a metadata object of a plurality of metadata objects, wherein the plurality of metadata objects are organized in a second hierarchical architecture; and

~~organize the runtime container and the metadata object at levels within a hierarchical architecture~~

allow each runtime container of the plurality of runtime containers in the first hierarchical architecture to invoke service components within the runtime container and provide state information and context information to the service components based on a meta object of the plurality of metadata objects in the second hierarchical architecture at a corresponding level.

44. (Original) The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:

extend the runtime container to inherit, provide, create and extend services, functionalities and properties of other runtime containers in the hierarchical architecture via inheritance mechanisms.

45. (Original) The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:

extend the metadata object to inherit properties, methods and interfaces of other metadata objects in the hierarchical architecture via inheritance mechanisms.

46. (Original) The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:

organize the runtime container and the metadata object in duality, wherein a container at one level in a hierarchical architecture is capable of accessing a metadata object at the same level in the hierarchical architecture.

47. (Original) The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:

create new types of runtime containers, or customize existing containers with incremental features.

48. (Original) The machine readable medium of claim 43, further comprising instructions that when executed cause the system to:

create new levels in the hierarchical architecture for the runtime container and the metadata object.

49. (Currently Amended) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

communicate with an invocation component and external clients via a routing and event handling component;

receive requests from the routing and event handling component, dispatch said requests to a service component within a runtime container of a plurality of runtime containers, wherein the plurality of runtime containers are organized in a first hierarchical architecture, and manage the returned responses via the invocation component;

process ~~processing~~ requests from the invocation component and producing responses back to the invocation component via the service component within the runtime container;

obtain ~~obtaining~~ state information from a nonvolatile storage and providing such

information to the invocation component and the runtime container;

obtain ~~obtaining~~ information from a metadata object of a plurality of metadata objects and providing such information to the invocation component and the runtime container, wherein the plurality of meta objects are organized in a second hierarchical architecture; ~~and~~

communicate ~~communicating~~ with the runtime container and external services; and

allow each runtime container of the plurality of runtime containers in the first hierarchical architecture to invoke service components within the runtime container and provide state information and context information to the service components based on a meta object of the plurality of metadata objects in the second hierarchical architecture at a corresponding level.

50. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

communicate with the invocation component using a uniform or standardized protocol.

51. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

create the service component in the form of Java Beans.

52. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

perform either pre-processing requests or post-processing of responses sent to or returned from the service component.

53. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

transparently map the service component to a more complex set of components to generate and deploy applications under runtime environment.

54. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

specify the service component declaratively and programmatically, as well as provide a model for declarative configuration override of the component at application deployment time.

55. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

locate, manage and persist state information, wherein the physical mechanism of doing so is transparent.

56. The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

expose component-level application services using the context information.

57. (Original) The machine readable medium of claim 49, further comprising instructions that when executed cause the system to:

provide a simplified and common interaction model to communicate with the external services.

58. (Original) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

manage communications between the runtime container and external entities using common or uniform protocols via at least one servlet;

monitor incoming communication at the servlet;

communicate with one or more servlets, determine which components to invoke, dispatch requests requiring asynchronous processing to a queue, and dispatch requests requiring synchronous processing directly to a stateful or a stateless component;

store asynchronous requests via the queue;

receive requests from the queue, determine which components to invoke, and dispatch requests requiring synchronous processing directly to a stateful or a stateless component;

process stateless requests via at least one stateless component; and

process stateful requests via at least one said stateful component.

59. (Original) The machine readable medium of claim 58, further comprising instructions that when executed cause the system to:

communicate in TCP/IP, HTTP, SOAP, XML, and other application-specific protocols.

60. (Original) The machine readable medium of claim 58, further comprising instructions that when executed cause the system to:

implement the stateless or stateful component using Java programming language in the form of Java Beans.

61. (Original) The machine readable medium of claim 58, wherein:

the stateless component is capable of at least one of the following:

deriving context information from the metadata;

containing an arbitrary amount of code for processing logic;

calling other stateless components within the container; and

utilizing synchronous or asynchronous controls to communicate with external services.

62. (Original) The machine readable medium of claim 58, wherein:

the stateful component is capable of at least one of the following:

deriving context information from the metadata;

retrieving state information from nonvolatile storage through a state management component;

containing an arbitrary amount of code for processing logic;

calling other stateless or stateful components within the container; and

utilizing synchronous or asynchronous controls to communicate with external services.

63. (Canceled).

64. (Canceled).